European Sixth Framework Network of Excellence FP6-2004-IST-026854-NoE

Deliverable D2.2 Virtual Laboratory Integration Report

The EMANICS Consortium

Caisse des Dépôts et Consignations, CDC, France
Institut National de Recherche en Informatique et Automatique, INRIA, France
University of Twente, UT, The Netherlands
Imperial College, IC, UK
International University Bremen, IUB, Germany
KTH Royal Institute of Technology, KTH, Sweden
Oslo University College, HIO, Norway
Universitat Politecnica de Catalunya, UPC, Spain
University of Federal Armed Forces Munich, CETIM, Germany
Poznan Supercomputing and Networking Center, PSNC, Poland
University of Zürich, UniZH, Switzerland
Ludwig-Maximilian University Munich, LMU, Germany
University of Surrey, UniS, UK
University of Pitesti, UniP, Romania

For more information on this document or the EMANICS Project, please contact:

Dr. Olivier Festor
Technopole de Nancy-Brabois — Campus scientifique
615, rue de Jardin Botanique — B.P. 101
F—54600 Villers Les Nancy Cedex
France
Phone: +33 383 59 30 66
Fax: +33 383 41 30 79
E-mail: <olivier.festor@loria.fr>

Document Control

Title:	Virtual Laboratory Integration Report
Type:	Public
Editor(s):	Jürgen Schönwälder, Ha Manh Tran
E-mail:	j.schoenwaelder@iu-bremen.de
Author(s):	WP2 Partners
Doc ID:	D2.2.doc

AMENDMENT HISTORY

Version	Date	Author	Description/Comments
V0.1	Octorber 28, 2006	Jürgen Schönwälder	Initial version
V0.2	November 24, 2006	Ha Manh Tran	Add changes in structure
V0.3	December 16, 2006	Ha Manh Tran	Add missing sections from the partners
V0.4	December 29, 2006	Jürgen Schönwälder	Added more sections, improved the presentation

Table of Contents

(This page is left blank intentionally.)

1 Executive Summary

This second "Virtual Laboratory Integration Report" presents the on-going activities of the virtual laboratory and common testbeds work package (WP2). The work package currently entertains four activities:

1. Creation and maintenance of a VoIP management testbed (VOIP)
2. Network management trace collection and analysis (TRACE)
3. Trace collection for network replay (REPLAY)
4. Resource usage data collection (ABLOMERS)

A detailed description of these activities including the description of the role of the WP2 partners can be found in the first "Virtual Laboratory and Integration Report". This report describes the progress made on these activities during the last six-month period. It emphasizes the successful construction of the VoIP testbed (VOIP) and the development of trace analysis tools and the creation of data trace repositories (TRACE) and the resource usage data collection project (ABLOMERS). A more detailed report of the trace collection for network replay (ABLOMERS) project will be provided in the next "Virtual Laboratory Integration Report".

In general terms, work package two has made significant progress to achieve closer collaboration. The construction of the VoIP testbed in particular led to close direct collaboration between the various partners involved. Some of the trace collection activities face issues with privacy concerns and more discussion within this work package is needed to figure out how to deal with non-disclosure agreements and the sharing of data in general within the EMANICS Network of Excellence, especially in those cases where the data sources is not an EMANICS member.

1 Introduction

The objectives of this work package are the integration of existing laboratories, the establishment of collaboration environment and the creation and maintenance of trace repositories for research and educational purposes. The first and second objectives are highlighted by constructing a Voice over IP (VoIP) testbed and sharing trace repositories among partners, whereas developing various tools for collecting and analyzing traces and establishing trace repositories emphasize the third objective.

The VOIP project aims at implementing a VoIP communication testbed over different partner sites. This testbed not only forms an intriguing platform for research activities but also connects partners for collaborative work. The project is divided into three phases. The initial phase has been completed and successfully brings a basic testbed into function. The last two phases are in progress and aim at upgrading the testbed security and expanding to more partners. The testbed currently comprises six partners.

The TRACE project involves the collection and analysis and network management traces. Six partners of this project are responsible for developing new tools for collecting and analyzing traces, and sharing trace repositories among partners. The status of this project can be epitomized by preliminary achievements. A tool called *snmpdump* developed in cooperation between UT and IUB allows partners to prepare collected SNMP traces for subsequent analysis and exchange. The tool supports multiple output formats and provides anonymization and filtering mechanisms. Traces are collected from different production networks, mainly from partner institutions and affiliated organizations, then stored and shared by partner repositories. Analyzing traces is an important part of this project. Partners like UT, IUB, INRIA use their own analysis tools for analyzing traces. LMU works on the model of distributed trace analysis using Grid technology. The construction of a demonstrator based on grid middleware in collaboration with CETIM is in progress.

The last two activities, REPLAY and ABLOMERS, deal with concrete applications that use specific network traces. The former (REPLAY) utilizes existing traces for network replay that allows researchers to evaluate distributed monitoring algorithm on real-world scenarios. This is a small project in terms of participants and EMANICS support and a more detailed report will be provided in the next deliverable. The later project (ABLOMERS) explores resource usage data for learning and evaluating load-balancing algorithms on distributed systems. It builds up a set of resource monitoring agents on different machines. These agents then report statistical resource availability data that is used for evaluation. The preliminary evaluation results disclose that monitoring agents remain stable during performance test period and consume less computational resources in their hosting nodes

1.1 Purpose of the Document

The first phase of this work package lasts eighteen months and contains three deliverables. The first deliverable submitted in June 2006 focuses on describing and integrating the existing lab infrastructure of the various partners and formulating four activities based on the basis of these infrastructures. This second deliverable reports the progress made implementing these activities which cover three main aspects. The first aspect described in section 2 concerns the formation of the VoIP testbed enabling a number of research activities on management and security. The second aspect is related to the development of network management trace analysis tools and the trace collection of different production networks. Sections 3 and 4 cover this aspect. The collaboration of partners in each project is summarized in section 5. Section 6 concludes this deliverable.

2 VoIP Management Testbed (VOIP)

2.1 Introduction

The number of Voice over IP networks is spreading world-wide. While this service is already essential for many individuals and enterprises (both SMEs and large companies), its management becomes crucial. To meet the real challenges for the management plane of VoIP networks, EMANICS partners have established a collaborative VoIP testbed. The main aim of this testbed is to provide the resources for the research community with a large scale and practical network to carry out various studies, analysis and implementation.

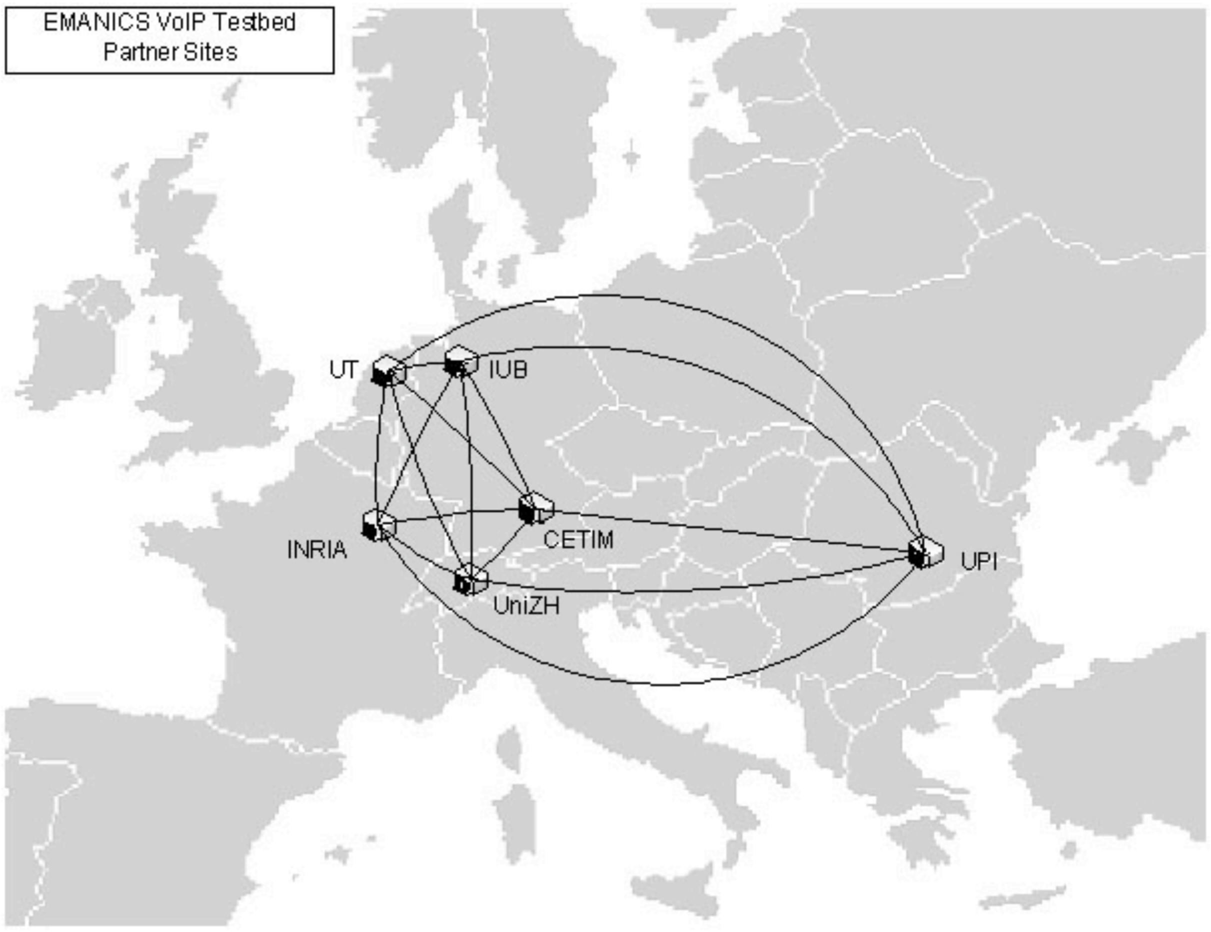


Fig 3.1 Collaborative VoIP Testbed with the Partner Sites

The testbed that is spread over the Internet and maintained by the partners provides us the virtue of a real network which gives a true value for the research work.

During the initial phase, the testbed has been setup using homogeneous core components. The infrastructure is based on the open source Asterisk [1] software and is fully functional. On the end systems part, the components are highly heterogeneous. Phase two of the deployment has started. It consists in integrating OpenSER [2] (a SIP [3]

Server) together with RADIUS servers in the testbed. The partners participating in the construction of the VoIP testbed are INRIA, CETIM, UPI, UniZH, IUB, and UT.

2.2 Testbed Schema and Specifications

The testbed is designed and implemented in different phases. This helps in understanding the state of art of the various technologies and their complexities in integration.

2.2.1 Phase I

The main objective of the first phase of the VoIP testbed implementation was concentrated towards building a fully operational VoIP network between the partner sites. In this phase, we have installed the network components and devices (both software and hardware) that are needed to have a basic VoIP network.

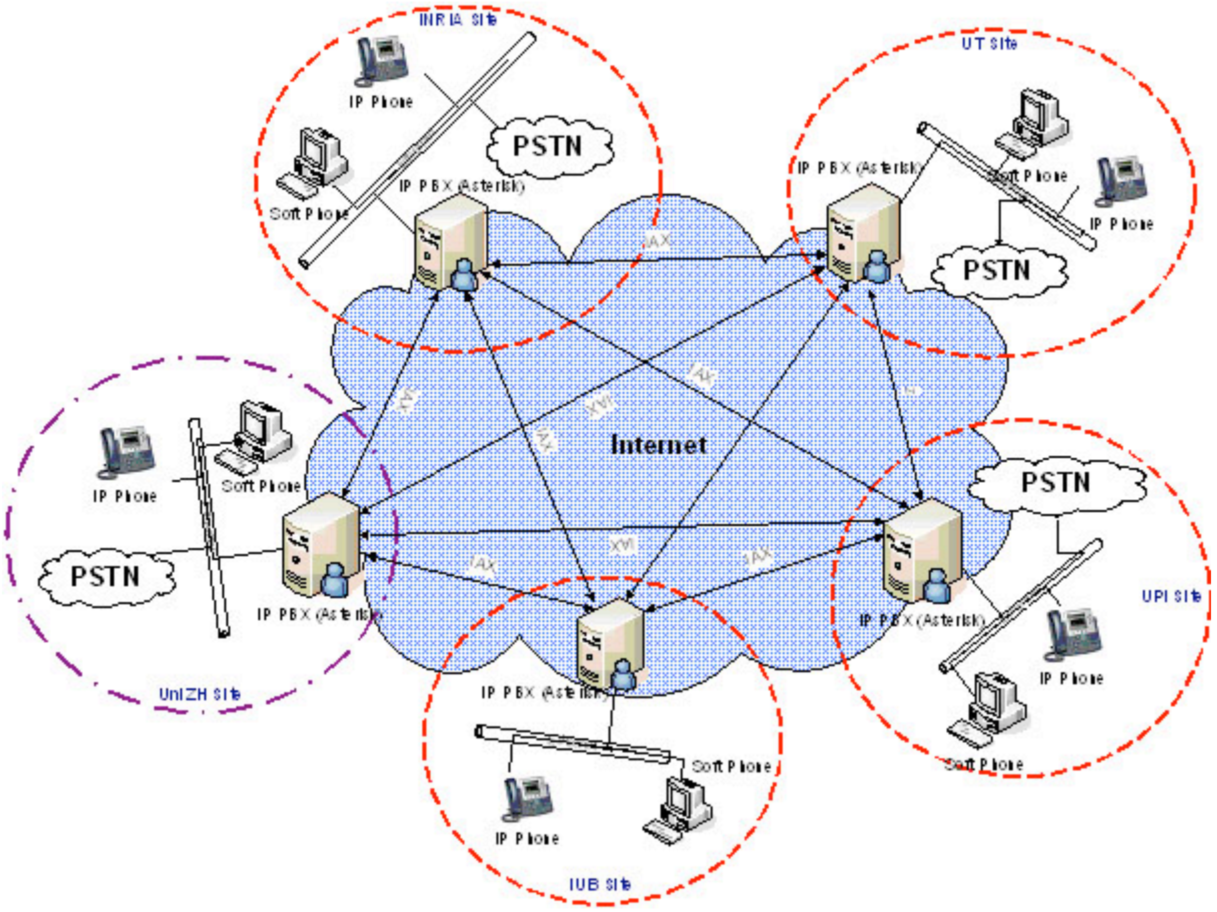


Figure 3.2 VoIP Testbed Layout of Phase I

The platforms consist of ordinary PC hardware running the Asterisk open source VoIP server under the Linux operating system. The participating partners integrated various VoIP phones, ranging from pure softphones such as *kphone*, *twinkle*, or *xLite* to hardware VoIP phones such as Cisco and Grandstream VoIP phones.

	System	IP PBX	Server IP	Users
INRIA	Linux	Asterisk	152.81.114.222	Olivier, Radu, Bala, Humberto, Mohamed

UT	Linux	Asterisk	-	Aiko
IUB	Linux	Asterisk	212.201.49.4	Jürgen, Ha
UPI	Linux	Asterisk	194.102.70.8	Florin, Luminita
UniZH	Linux	Asterisk	192.41.135.197	Fabian, Gregor, David

Table 3.1 The platform specification of the partner sites

Dial Plan Specifications

The dial plan provides us the architectural connection between the different sites. The following is the extension configuration setup implemented in the different sites to have interconnection between the sites. Note that local PBX settings can be defined by the partners or if needed we could define a common setup for local extensions.

	INRIA Site	**UT Site**	**IUB Site**	**UPI Site**	**UniZH Site**
	Routing No. - Extension	Routing No. - Extension	Routing No. - Extension	Routing No. - Extension	Routing No. - Extension
User 1	400-101	500-101	600-101	700-101	900-101
User 2	400-102	500-102	600-102	700-102	900-102
User 3	400-103	500-103	600-103	700-103	900-103
User 4	400-104	500-104	600-104	700-104	900-104
User 5	400-105	500-105	600-105	700-105	900-105

Table 3.3 The dial plan specification of the partner sites

The Asterisk server has various configuration files that need to be configured. The main configuration files that were utilized for the phase I implementations are the following:

extensions.conf	- Configuration file for the dial plan
sip.conf	- Configuration file for SIP channels
iax.conf	- Configuration file for IAX channels
voicemail.conf	- Configuration file for voicemail
meetme.conf	- Configuration file for conference calls

The essential task of partners was to install and configure a SIP router and Asterisk PBX. Depending on the infrastructures and usage purposes, partners may set up more components. The configuration of UniZH and IUB sites are examples. While IUB sets up a VoIP lab for the testbed and the VoIP lab session, UniZH integrates its own VoIP gateway with a campus PSTN connection into the testbed.

At the IUB site, the VoIP lab contains an Asterisk PBX with SIP and IAX channel support and some SIP softphones and hardphones. The Asterisk PBX operating on a Xen virtual Debian GNU/Linux machine allows basic functions described in the VoIP specification to work successfully except for the conference call function. Figures 3.3 plots the basic VoIP network of IUB site.

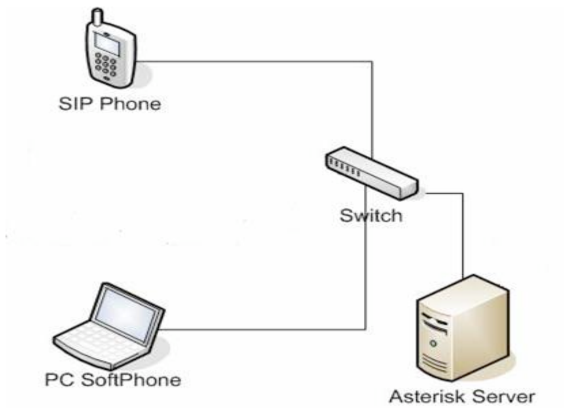


Figure 3.3 The VoIP network of IUB site

At the UniZH site, the Asterisk PBX has been installed on a dedicated Dell PowerEdge 850 Server (Pentium 4 3.6GHz, 1 GB RAM), which is directly connected to the Swiss research network SWITCH. Peerings with the IAX protocol have been set up with the other VoIP testbed participants UT, IUB, UPI and INRIA. Test calls were successfully placed. The local dial plan has been adapted according to the VoIP specification allowing internal and external calls. Currently there are four local users configured. Several types of software and hardware clients are used: Cisco 7940 Phone, Grandstream GXP-2000 and the xLite software. In addition to the SIP support, an H323 was installed, allowing the use of wider variety of clients.

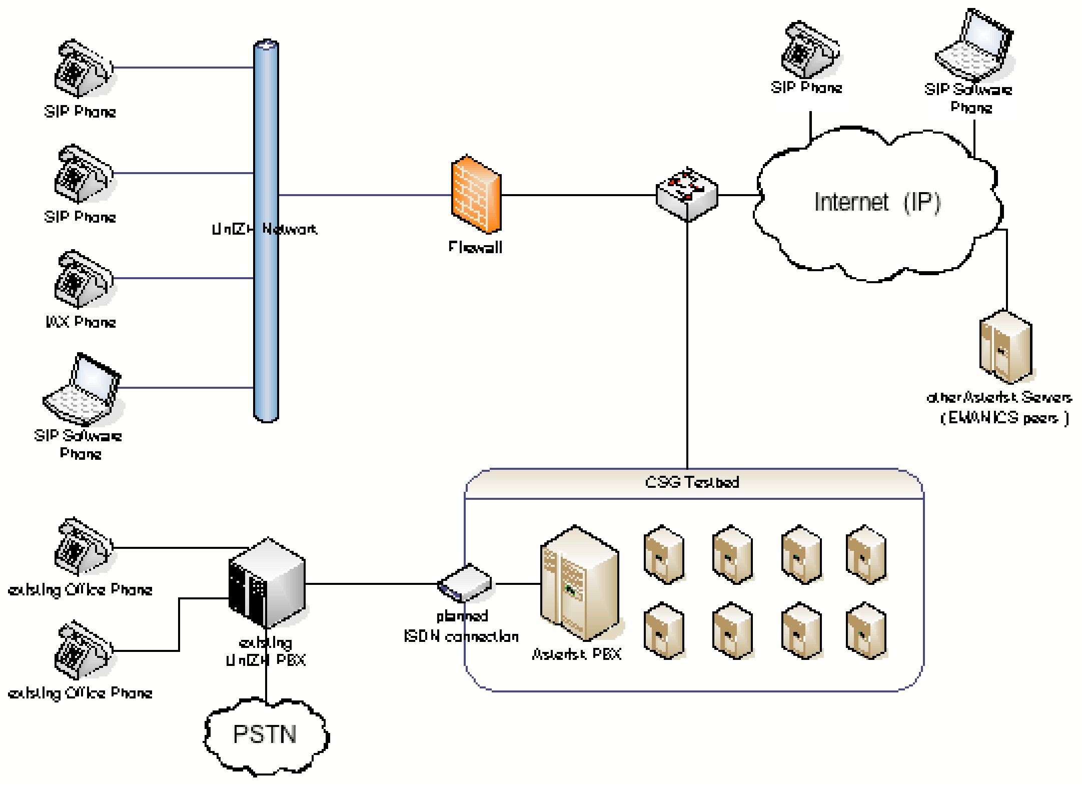


Figure 3.4 The VoIP network of UniZH site

The Asterisk MeetMe was set up to host conference calls. Each user is given a voice-mail where callers can leave messages for the user if he is unreachable. The user then receives the message as an attachment to his e-mail address. Instead of dialing the extensions, the users are also reachable by entering their e-mail address on compatible clients. The logging of call detail records is left at its default configuration, making entries for each call in a daily logfile and recording original, destination, duration and the date and time the call was placed. Figure 3.4 depicts the VoIP network of the UniZH site.

2.2.2 Phase II

Having implemented the basic VoIP network in the first phase with Asterisk, we started to expand the network with a dedicated SIP server (*OpenSER*). This SIP server will act as a SIP proxy and will provide centralized authentication for SIP clients in the testbed. Apart from this we also expanded our network by integrating one more site with our partners namely CETIM. CETIM has been assigned the dial plan prefix 800.

The details of the specification changes and scenarios will be adopted depending on the requirements. This is a major phase in which the evolving VoIP technologies like SIP will be utilized to the core to provide maximum services that are available to the research community, thus enabling to understand the different architectures.

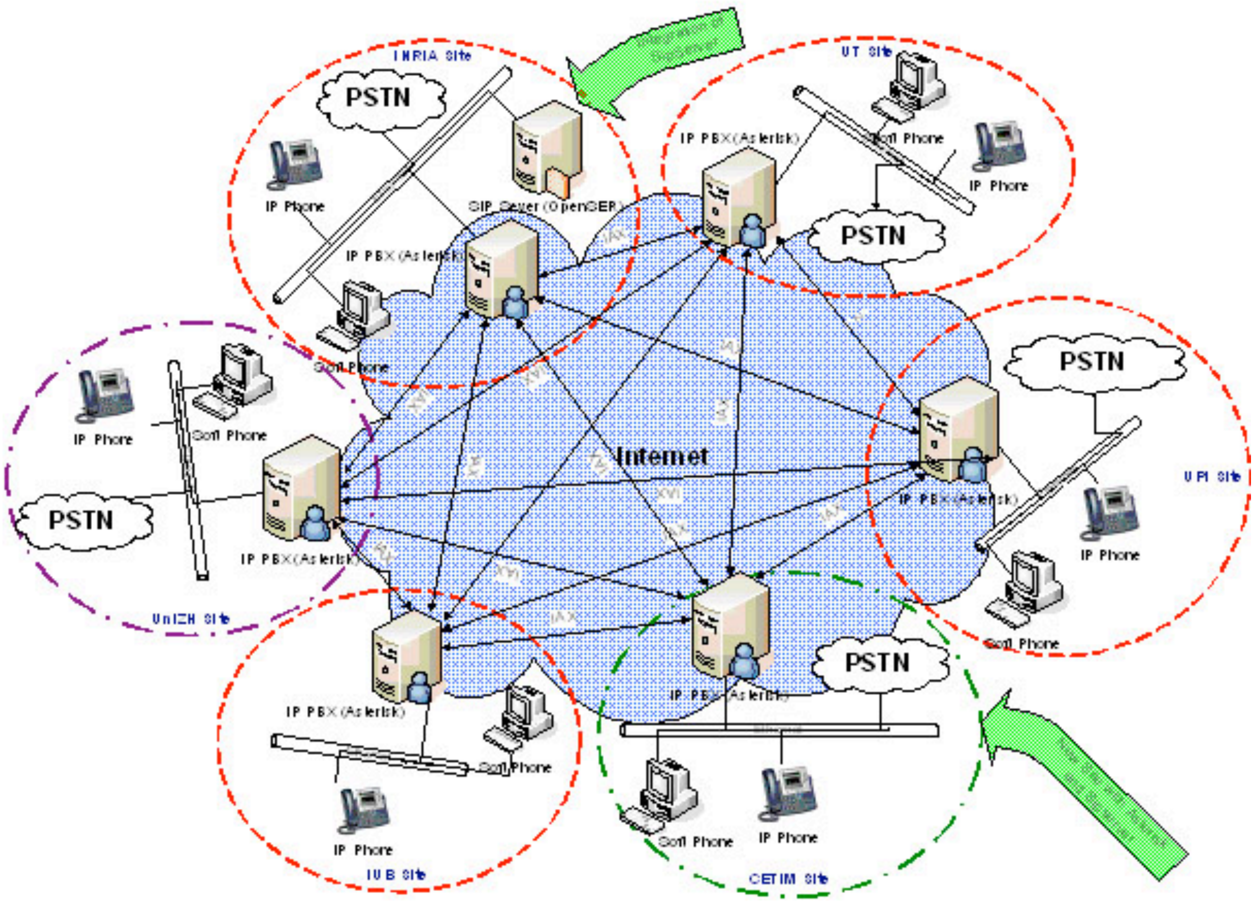


Figure 3.5 VoIP Testbed Layout of Phase II

2.2.3 Phase III

During the third phase, we will integrate RADIUS servers to have a centralized administration of the VoIP user accounts. This enables the testbed to have a secure administration and management of the network.

The details of the specification changes and scenarios will be decided after the tests carried out from the implementation of phase II and depending on the requirements.

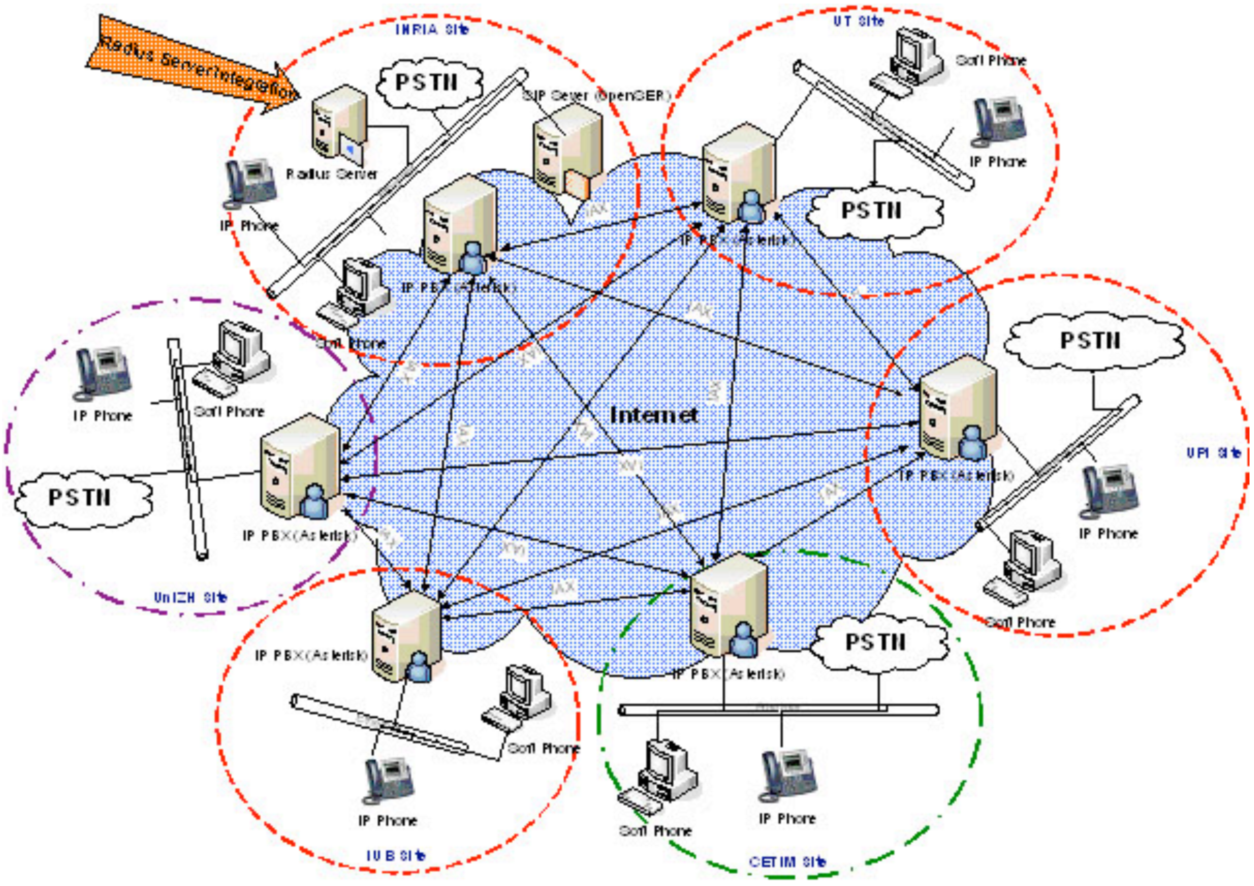


Figure 3.6 VoIP Testbed Layout of Phase III

2.3 Testbed Activities

There are multiple activities already initiated over the testbed. Some of the important and major activities that are carried out are:

VoIP Network Management: The management of the VoIP network is quite different from that of the traditional networks because of its need for dedicated bandwidth, the real-time nature and the evolving new technologies. Therefore it is important to understand and have better tools to meet these challenges. INRIA is there working on VoIP bots that can be used to both feed the network and monitor its activity.

Security Management: Security is still in the beginning stage for VoIP services. With already many problems emerging in the context of security for these networks, this will pose a big challenge in the future. Therefore it is important to carryout different experiments that will provide new solutions as quickly as possible. This activity is currently carried out at INRIA, which uses the testbed to validate its VoIP assessment models and algorithms.

Peer-to-Peer Management: The evolution of the distributed networks and peer to peer networks has made technological break through and has quite a good influence on the VoIP network as well. Therefore it is also important to study new paradigms for such networks. This is currently done in the CETIM premises with DUNDi [4] and will be integrated into the testbed framework.

2.4 Scenarios and Tests Accomplished

There are various scenarios that are conducted and accomplished in the testbed depending on the activity requirements. Some of the basic tests that were conducted are described below:

Function	**Case Description**
Internal calls	**Case 1:** SIP Client 1 → Asterisk → SIP Client 2 *Example : 101 → Asterisk → 102*
Calling the partner sites	**Case 1:** SIP Client 1 → Asterisk → IAX Channel → Asterisk → SIP Client 2 *Example : 400101 → Asterisk → IAX Channel → Asterisk → 700101*
Voicemail	**Case 1:** SIP Client 1 → Asterisk → IAX Channel → Asterisk → SIP Client 2 (Busy)→ Voicemail *Example: Calling from 400101 → 700101* *400101 → Inria-asterisk-server → IAX Channel → upi-asterisk server → 700101* **Case 2:** SIP Client1 → Asterisk → IAX Channel → Asterisk → SIP Client2 (No answer)→Voicemail *Example: Calling from 400101 → 700101* *400101 → Inria-asterisk-server → IAX Channel → upi-asterisk server → 700101 (no answer)→ Voicemail* **Case 3:** SIP Client1 → Asterisk → IAX Channel → Asterisk → SIP Client2 (Default)→Voicemail *Example: Calling from 400101 → 700101* *400101 → Inria-asterisk-server → IAX Channel → upi-asterisk server → 700101 (busy)→ Voicemail*
Conference calls	**Case 1: Calling an extension to join a conference (Local to each site)** Create a conference I.D (ex. 212) and add *meetme* application to dial plan (ex. Add extension 18 for conference) *SIP Client 1 → dial Ext 18 → enter conference I.D 211 → Joined the conference* *SIP Client 2 → dial Ext 18 → enter conference I.D 211 → Joined the*

	conference **Case 2: Create dedicated extension for conference** In this case we have direct conference extensions (999, 998,..). These are dedicated conference extension numbers. *SIP Client 1 → dial 999 → <join the conference>←dial 999 ← SIP Client 2* **Case 3: Secure conference rooms** In this case we add a password for the conference rooms (ex., 995 with password.) *SIP Client 1 → dial 995 → Enter password xxxxx → joined the conference* **Case 4 : Conference for EMANICS Project** In this case all the partners can join a conference hosted in the INRIA site for the EMANICS project. *SIP Client 1 → dial 777 →IAXchannel →Asterisk (INRIA) → Enter password xxxxx→ joined the conference* This scenario will also be used to host EMANICS conference at each partner site rotationally.

2.5 Summary

A VoIP testbed has been established between partner sites located in diverse locations in Europe. It provides an infrastructure to carry out VoIP research concerned with the management or security aspects of VoIP networks. The testbed integrates several open source software packages and heterogeneous hardware platforms and serves as a real world environment that encourages academia in advancing their research by taking advantage of a European testbed.

The activities and experiments that were accomplished so far in the testbed helped in understanding the state of art technologies and their complexities. In a further step to organize assorted experiments and research activities, the implementation and expansion needs will be discussed during the WP2 meeting to be held at the General Assembly in January 2007.

3 Network Management Trace Collection and Analysis (TRACE)

3.1 Introduction

The collection of network management traffic traces requires the support of network operators. On the technical side, good capturing points have to be defined and configured. On the non-technical side, an agreement has to be defined under which data can be shared and results published.

It is in usually not possible to make network management traces which contain complete messages openly available since they contain sensitive information. In some cases, we have settled on agreements, which make data available to specific researchers for research purposes while in other cases the operators involved keep the data and instead run our analysis software and provide the aggregated results back to us.

3.1.1 Sharing Traces

In the first approach, an operator collects traces and subsequently makes them available to researchers for further processing. Since traces contain data of different levels of sensitivity, operators typically want to exercise some control over the data that is given to researchers. Good examples are SNMP community strings, which are often used as clear-text passwords and hence should be removed. Since removing such data in binary *pcap* files which contain BER encoded SNMP messages is technically non-trivial and also difficult to verify, there is a need for a human and machine readable representation which makes it easier to (a) identify data to be removed, (b) to actually remove the identified sensitive data, and (c) to verify that the removal was successful.

Next to the filtering of highly sensitive data, some operators also prefer to have data anonymized so that the risk of leaking sensitive information is reduced. Even though strong anonymization is difficult to achieve, it is still often considered useful to achieve at least a level of pseudonymization as a second safety measure to complement a non-disclosure agreement. Anonymization requires applying a filter-in principle where only data for which an appropriate anonymization function exists is retained in a trace. Thus, anonymization can reduce the usefulness of the traces for researchers and it therefore requires some careful planning on the side of the operator involved. Furthermore, the development of suitable anonymization functions is still an ongoing research topic and hence the required software tools are experimental and changing rapidly.

3.1.2 Sharing Analysis Software

The alternative approach to sharing traces is to share the analysis software and to ask operators who own the traces to execute the analysis software on behalf of researchers. Of course, operators who execute analysis software provided by researchers have to trust the software and check the results against their privacy requirements. As a consequence, analysis software should be open, portable, and reasonably well documented. Analysis software should be provided in a way which makes it possible for operators to verify that the code does not contain any unwanted features. This also implies a pragmatic selection of programming languages so that programs are likely to be understood by the operator community.

Experience so far tells us that a combination of both approaches usually works reasonably well. In such cases a researcher obtains potentially filtered and anonymized traces from an operator (typically legally covered by an agreement between the operator and the trace analyst) and the researcher executes analysis software provided by other researchers interested in the trace (assuming that is covered by the agreement).

3.1.3 Intermediate Formats

To support the approaches mentioned above, two intermediate formats for SNMP traces have been developed [5]. The first one is an XML format which is intended as a human and machine readable exchange format which is capable to retain all information found in BER encoded SNMP messages. This format is relatively verbose (traces in XML format are typically a factor 7 larger than the original pcap trace file), but this can be mitigated by compression. A large number of tools does exist to process XML files and so ad-hoc transformations are feasible. However, our experience is that many XML tools do not scale very well to large data sets.

The second intermediate format is a simple CSV (comma separated values) format which only retains the most essential information. It turns out that processing CSV files is usually much faster, especially since many line-oriented tools can be applied directly. The downside of the CSV format is that it is not very flexible and changes in the CSV format typically break tools in surprising ways. Hence it is crucial to get this format right and to keep it stable.

3.2 Tools

Raw network traces are typically captured using *tcpdump* and stored in *pcap* format. Hence, any generic tools that can process raw *pcap* files can be used to split and merge raw traffic trace files. In order to analyze the traces, we have developed additional tools. Since traces may become quite large, tools should be relatively efficient and fast.

3.2.1 Conversion Tool *snmpdump*

A new tool called snmpdump accomplishes the conversion of raw pcap traces containing SNMP messages into intermediate formats. It reads raw *pcap* files as input and produces traces in XML or CSV format as output. Since the XML format retains all information, it is also supported as an input format. This allows using *snmpdump* as a filter. In addition, *snmpdump* accepts CSV format as input even though this format does not retain all information.

Parsing: To extract SNMP messages from raw pcap files, it is necessary to (a) deal with fragmentation by reassembling fragmented datagrams and (b) decode the native BER format into an internal representation. For packet reassembly, we used the *libnids* library which essentially contains a user-space version of some portions of the Linux TCP/IP networking code. For the BER decoding, we modified the SNMP packet decoder shipped with *tcpdump* to construct an in memory representation of an SNMP message.

To parse data stored in the intermediate XML format, we use the *libxml* reader API. The data provided by the generic XML parser, which automatically checks well-formedness, is used to generate an in memory representation. Although not surprising, it should be noted that parsing raw *pcap* files is significantly faster and hence *pcap* remains a good choice for processing large trace files if no filtering or anonymization is needed.

The CSV parser is a straight-forward implementation which reads a line, tokenizes it and then generates an in memory representation of the message. The CSV parser is handy when existing CSV files have to processed further.

Data Representation and Control Flow: SNMP messages are represented internally using a nested data structure, which can represent the different SNMP message formats. Every data member which carries the value of a field of an SNMP message has associated attributes. These attributes control memory management and indicate whether a value is present. By adding these attributes to all data members, we are able to pass a decoded SNMP message through several stages of a processing pipeline until the message is finally serialized into one of the output formats. The drivers, which produce different output formats, adhere to a common interface, which makes the implementation extensible. The overall data flow within *snmpdump* is shown in Figure 4.1.

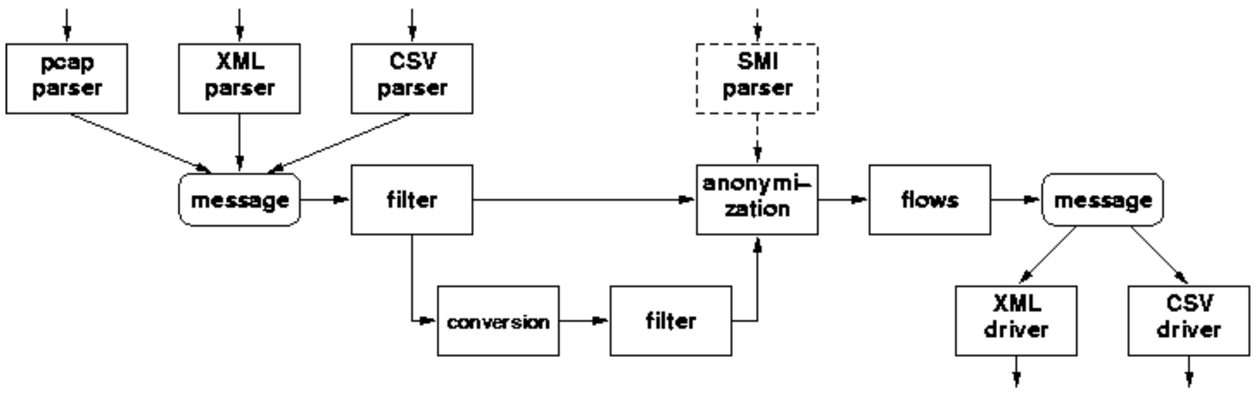


Figure 4.1 Data flow within the snmpdump data conversion tool

Filtering: The filter module of *snmpdump* is responsible to filter out message fields that should be suppressed, for instance because specific sensitive data must be removed. The message fields that should be suppressed are selected using a regular expression and the suppression essentially changes the attributes of the selected message fields. As a safety measure, the data stored in filtered message fields is cleared or set to some standard “null” value, just in case some other code forgets to check the attributes when accessing message fields.

Conversion: The format of the payload of SNMP messages changed when the second version of SNMP was introduced. In particular, the format of unconfirmed traps was changed and harmonized. The coexistence specification [6] defines a conversion procedure which allows traps in the old format to be translated into the new format and back. The conversion module implements this conversion procedure in order to provide a uniform interface. Note that the conversion module can be bypassed if no conversion is desirable. If conversion has been performed, it is necessary to call the filter module again since the conversion might have filled message fields with values, which were not present before.

Anonymization: The anonymization module is responsible for anonymizing message fields. It makes use of a reusable anonymization library called *libanon* [7]. The library provides anonymization functions for standard data types such as signed / unsigned integers and octet strings as well as specific functions for MAC addresses or IP addresses. The anonymization functions support a lexicographic-order-preserving mode in order to preserve SNMP's lexicographic-order property of instance identifiers. A more detailed description of prefix- and lexicographic-order-preserving IP address anonymization can be found in [7].

In order to select the anonymization function for a given message field, it is necessary to have some context information, such as the object descriptor or the object's type name. The anonymization module therefore looks up data definitions by calling the *libsmi* library, an embeddable MIB parser library. Note that these lookups are only performed if anonymization has been requested and the values in question are actually present. The selection of the anonymization function to apply for a given object or a given data type is runtime configurable.

Flow Identification: Large traces, which may contain interactions of several managers with hundreds of agents, usually have to be broken into more manageable pieces. A natural choice is to split a combined trace into several traces, each one representing a message flow. An SNMP message flow is defined as all messages between a source and destination address pair which belong to a command generator (CG) / command responder (CR) relationship or a notification originator (NO) / notification receiver (NR) relationship.

The above definition deliberately does not consider port numbers. The reasons are twofold: First, most managed devices include just a single SNMP agent. Even if multiple agents are present, either subagent protocols or proxy mechanisms are usually used to hide this. Even if a device has multiple totally independent SNMP agents, we still consider the device a single logically managed device. Second, many managing systems make heavy use of dynamically allocated port numbers which can change frequently and thus would cause lots of unrelated flows to be generated, even though all the flows are coming from a single management station.

The implementation of the flow identification module requires to deal with reordered messages and to associate responses (and reports) to prior requests since responses do not indicate whether they are send in response to a notification or a data retrieval operation.

3.2.2 Conversion Tool *rmidump*

This tool developed by the INRIA team assists performance tests of the JMX management framework. One metric of importance is the delay of a management request. One workload factor of importance is the number of requests injected per second. Currently we extract the request delays by analyzing application-level traces and we use network level traces obtained by the tool to check with accuracy the actual value the load injection for a given test. The *rmidump* utility consists of two components:

- A java library which reads network trace files (pcap format) and extracts Java Remote Method Protocol (JRMP) data and so allows easy trace of Java Remote Invocations (RMI). This library can be used to write network level analyzers of RMI calls latency for a given application for example.

- A plugin for the Wireshark[1] (former ethereal) network sniffer, which pins-out packets related to RMI/JRMP and deserializes return values or parameters as best as possible. This part is mostly dedicated to trouble shouting in java RMI applications.

3.3 Traces

The partners involved in this project have collected a number of traces. The following list provides a brief description. For more details about the traces and how to access them, please contact the partner directly.

3.3.1 University of Twente

UT has collected at four different locations: at the backbone of SURFnet, which is the research network provider within the Netherlands, at a University network, at a faculty network, and at a server-hosting provider. At the first location, trace collection started already in July 2005; at the other locations traces were collected in April and May 2006. At the first location, traffic was copied from the access router of the Network Operations Centre (NOC); at the second location, traces were collected from separate management VLANs. At location number four. data was collected by capturing all SNMP traffic originating from or destined to a specific network manager. In some cases UT could also capture other management traffic, which gave the possibility to compare SNMP usage to that of other management protocols. Files were stored in *pcap* format using the *tcpdump* program.

3.3.2 International University Bremen

IUB has collected SNMP traces mainly for testing purposes at the local networking lab, which is monitoring only a small server of devices and services. In addition, two traces have been collected at a regional network provider during April-June 2006. Additional SNMP traces have been received from a European and a Brazilian research network operator.

3.3.3 Poznan Supercomputing and Networking Center

PSNC has collected SNMP traces from the National Research Network within Poland from August to September 2006. Trace collection of the POZMAN metropolitan network and the PSNC LAN is in progress. Traces were anonymized using the *snmpdump* tool and verified prior to sharing then within the NoE

3.3.4 University of Federal Armed Forces Munich

CETIM has collected pcap header traces from University of Federal Armed Forces network link, which connects the residential building of the computer science students to the campus backbone network. Data was collected over a period of eight days and data was stored in *pcap* format using the *tcpdump* program. CETIM has also started discussing collection possibilities at the campus backbone and the campus' uplink to the DFN backbone X-WiN.

1 www.wireshark.org

3.3.5 Ludwig-Maximilian University Munich

LMU has installed the infrastructure to capture and anonymize NetFlow traces at the Leibniz computing center (LRZ). A trace repository server is set up at LMU providing roughly 700 GB of free storage for anonymized NetFlow traces. First NetFlow traces from the Munich backbone should be available to TRACE partners soon.

Table 4.2 provides an overview of the collected traces by the different partners.

Partner	Source	Type	Duration	Size
UT	national research network	SNMP trace	7 days	5.59 GB
UT	university network	SNMP trace	10 days	46.35 GB
UT	faculty network	SNMP trace	32 hours	1.08 GB
UT	server hosting provider	SNMP trace	4 hours	0.006 GB
IUB	regional network provider	SNMP trace	24 days	2.5 GB
IUB	national research network	SNMP trace	9 days	25 GB
IUB	national research network	SNMP trace	9 days	0.428 GB
IUB	laboratory network	SNMP trace	25 days	0.625 GB
PSNC	national research network	SNMP trace	33 days	2.39 GB
CETIM	university network	header trace	8 days	4.5 GB

Table 4.2 Overview of the traces collected by different partners

3.4 Procedures

This section proposes general procedures and rules that all partners involved in the trace collection and analysis project may agree to. It aims at the reduction of efforts and exceptions to allow a higher level of automation. At the same time, it tries to leave as much control as necessary to the trace providers to encourage them to share their trace.

To enable a quick start, procedures are described with a two step implementation in mind: In the first step, the goal is to get something running fast based on known and common tools. In a subsequent second step, a grid infrastructure should be established to support the trace analysis activities. Similarly, it is useful to start with individual trust relationships, with the aim to establish group trust later.

For the required metadata, we propose to start with semistructured formats, which allows flexibility by using verbal descriptions in natural language to work around having to predict all possibilities. Once we got a grip on what kind of information we actually need, more formal data structures can be introduce and any existing metadata in natural language can be converted to a more structured and more formal format manually due to its low amount.

The data formats are described here with basic semantics but without a proposal for explicit syntax. This has been done for the following reason: We assume that we should focus on what data is needed before discussing individual syntax of the data fields. The data fields in all data structures are filled in when the information is available. They can be updated at any time, so that changes in existing records can be made. This includes the references between the records.

We assume that a partner involved in a trace analysis project has a certain role. (Partners may have multiple roles but usually only one role for a given task at a specific point in time).

- A *Trace Provider* (TP) provides traces to other parties. A TP typically has access to larger production networks and typically has an operator background.
- A *Trace Analyst* (TA) likes to perform analysis tasks on traces. A TA typically has a research background.
- A *Tool Developer* (TD) creates tools for trace conversion, anonymization, analysis and so on. A TD may not be directly involved in trace collection or analysis, but he needs to receive feedback in order to improve the tools.

There are obviously a number of interactions between TPs, TAs, and TD. This section will later describe these interactions in more detail.

3.4.1 Trace Metadata Directory

The Trace Metadata Directory acts as a central information story. It contains metadata, i.e., data to describe traces, activities and results but no actual traces or results. In its simplest form, it contains records of the following four types listed in the table below. The specific data fields for each record type are specified later in this document.

CAPTURE REQUEST RECORD	request for trace captures
CAPTURE ANNOUNCEMENT RECORD	announcement of captured traces
ANALYSIS ACTIVITY RECORD	analysis activity on traces
ANALYSIS RESULTS RECORD	analysis results

The Trace Metadata director serves as central information directory and activity monitor for the trace project. It stimulates documentation in a central place. Large portions of content of reports on integration and coordination, activities and results – as input for management reports and WP2 deliverables – will later be derived from this information. It further stimulates cooperation and interaction among partners by making metadata and activity data available to everyone in TRACE (or WP2 or EMANICS).

3.4.2 Generating Captured Traces

There exist capture requests (grouped with intended analysis activities) and capture announcements. Either of them may be transmitted to the trace metadata directory independently from the presence of the other one. The structure of requests and announcements is similar to ease matching requests with announcements, either by humans or some software logic (later).

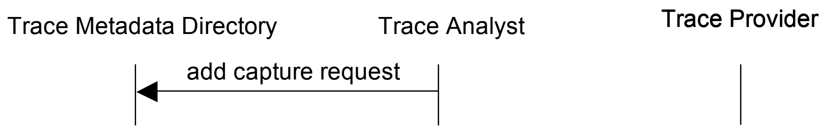
3.4.2.1 REQUEST_TRACE_CAPTURE

Informal description:

"We would like to analyze traces with the following properties (capture request) and therefore ask all partners to

capture relevant traces. [Optional, but encouraged] We want to perform this type of analysis on the data (analysis activity) (see next page)."

A trace analyst requests new traces with certain properties to be captured by some trace provider. The trace analyst should also file an additional analysis activity record to express what kind of analysis is intended to be run on the data. This description should give trace providers an imagination of the research activities on their data. Trace providers either poll the directory or subscribe to directory changes.



CAPTURE REQUEST RECORD

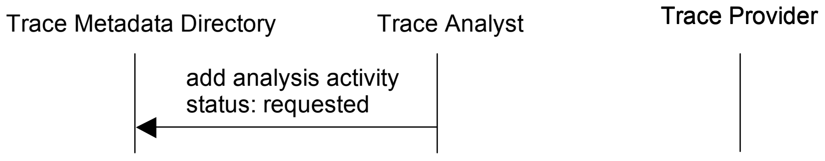
metadata	Id	identifier for one capture request
	trace analyst	name of EMANICS partner
	contact	name and e-mail address of a contact person
	request valid from ... until ...	
description of properties of requested capture	content	describe briefly what kind of traces you want, for a more detailled description of the kind of analysis that you intend to carry out, use an additional "analysis activity record"
	format	specify the format (see Implementation section for suggestions)
	environment	specify the environment, from which you would like traces
	capture time period	From what period do you want traces?
	total size, chunk size	requirements/wishes on minimal size/max size, chunk size
requirements	requirements on distribution and execution, e.g. JOB/COPY	JOB (provider/analyst) allowed/wanted/ required COPY (push/pull) allowed/wanted/required
	requirements on (non)-	e.g. a trace analyst may want to ana-

	anonymization/filtering	lyze the connection management in VoIP traffic and may require that all SIP communication for connection management remains unaltered, but the phone numbers and IP addresses can be anonymized via a bijective function, the actual voice data can be filtered if possible, specify anonymization/filtering in terms of an anonymization tool and its input parameters

3.4.2.2 SPECIFY_ANALYSIS_ACTIVITY

Informal description:

"We would like to perform this kind of analysis activity."



ANALYSIS ACTIVITY RECORD

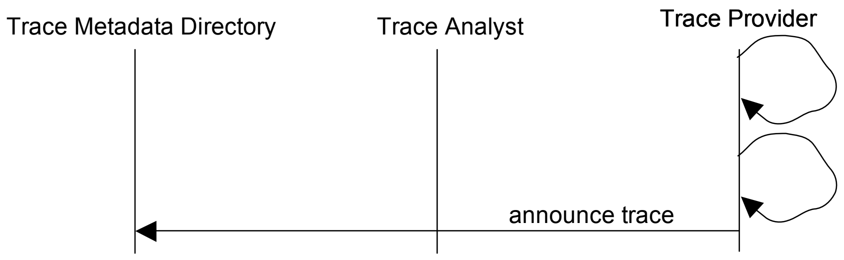
metadata	id	
	reference to CAPTURE REQUEST RECORD	
	reference to CAPTURE ANNOUNCEMENT RECORD	
execution	scripts to execute	for JOB/provider method: include analysis scripts and input parameters
	estimated time of completion / deadline	this data is a hint for humans, and later needed for a scheduler
	status: requested / running / completed	
description of analysis job	textual description of the intended analysis tasks	

3.4.2.3 ANNOUNCE_TRACE_CAPTURE

Informal description:

"We have captured a trace with the following properties (capture announcement). [Optional: With this trace, we try to provide data for this/these existing capture requests.]"

A trace provider announces the availability of a newly captured trace. If the new trace has been carried out to match an existing request, a reference in the analysis activity record should be added. Trace analysts either poll the directory or subscribe to directory changes.



CAPTURE ANNOUNCEMENT RECORD

metadata	id	identifier for one trace
	TRACE partner	EMANICS partner acting as trace provider, we need this for internal reports
	network	name of the network, where the trace was taken
	organization	name of the organization operating the network
	contact	name and e-mail address of a contact person
properties of captured trace	capture time period	when was the trace captured: start date and end date in ISO format
	format	format identifier/short description (see Implementation section for details)
	total size, chunk size	total size and (opt.) size of chunks, if the data is split into chunks
	content description	describe the content briefly, include a short description of the infrastructure as help for the trace provider
	periodicity	How often will you provide new data? e.g. monthly, weekly
	location and access	e.g. URL or path on a file server, include all information needed for other TRACE partners to access the data, e.g. how to gain accounts
requirements	requirements on (non)-	COPY (push/pull) allowed/wanted/required? JOB (provider/analyst) allowed/wanted/required?

	distribution and processing	redistribution of traces?
	availability	When is the data available to others? A capture for Aug 2006 may be available from Sep 15 to Dec 15 2006. Reason: Large traces may need to be processed before making them available and they may be too large to store them for a long time.
	requirements on anonymization/ filtering	What needs to be anonymized? How has the data been anonymized/filtered? if possible, specify anonymization/filtering in terms of an anonymization tool and its input parameters

3.4.3 Analyzing Captured Traces

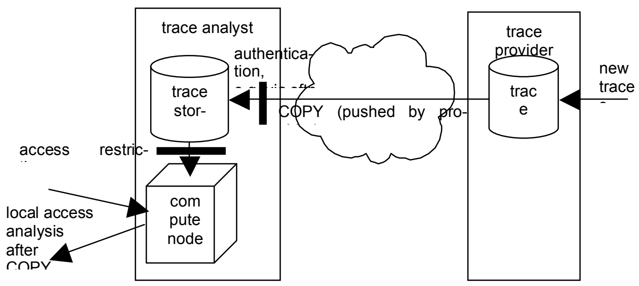
This section describes the specification and execution of trace analysis tasks.

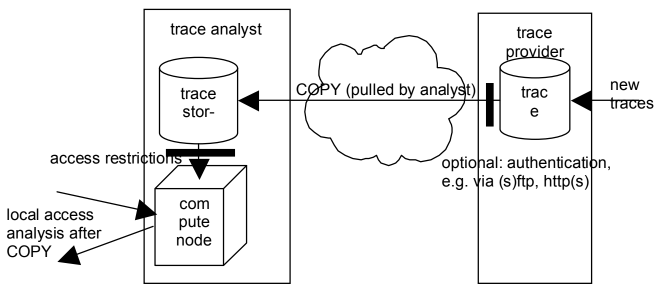
3.4.3.1 REQUEST_TRACE_COPY

Informal description:

"Dear trace provider, we have filed an analysis activity. We would like to receive a copy of a trace that you announced to run an anlysis job on the data locally."

The trace provider is free to decide whether to provide the trace in a push or pull fashion (a preference should have been specified in the capture announcement).

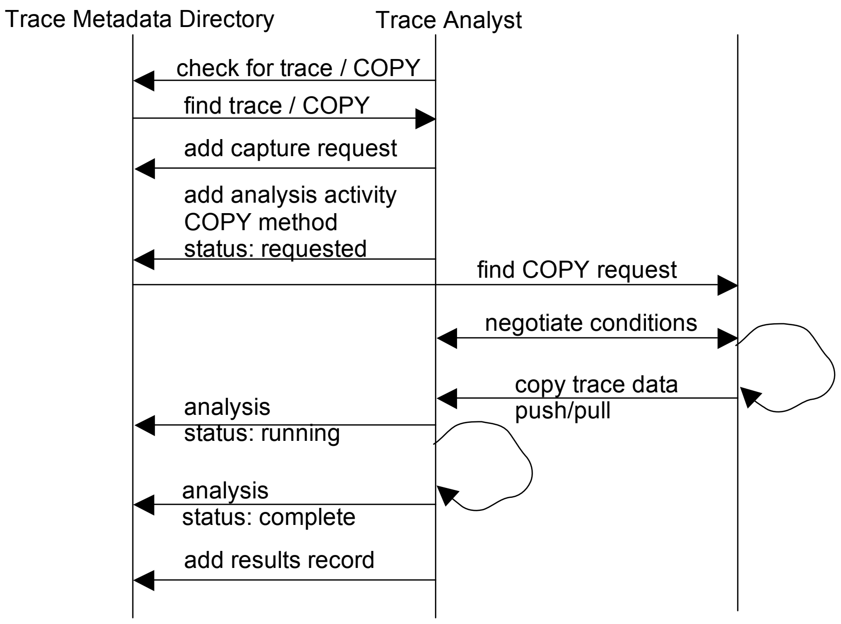




PUSH: The trace provider intiates the copying of data to a certain storage place specified by the analyst. This way the provider has full control on the transfer: the data in the trace repository cannot be downloaded without permission, the trace provider can enforce a certain type of transfer or a certain time (due to the large size of traces, and a potential influence on available bandwidth, night hours may be preferred). In addition, the trace provider can keep logs, to which other parties a copy was sent.

PULL: The trace may be accessible for download for non-sensitive data. The trace provider names a URL with the trace announcement, where other partners can download the trace. Authentication may be handled via credentials provided on request.

The trace analyst then runs the analysis on the local data copy.

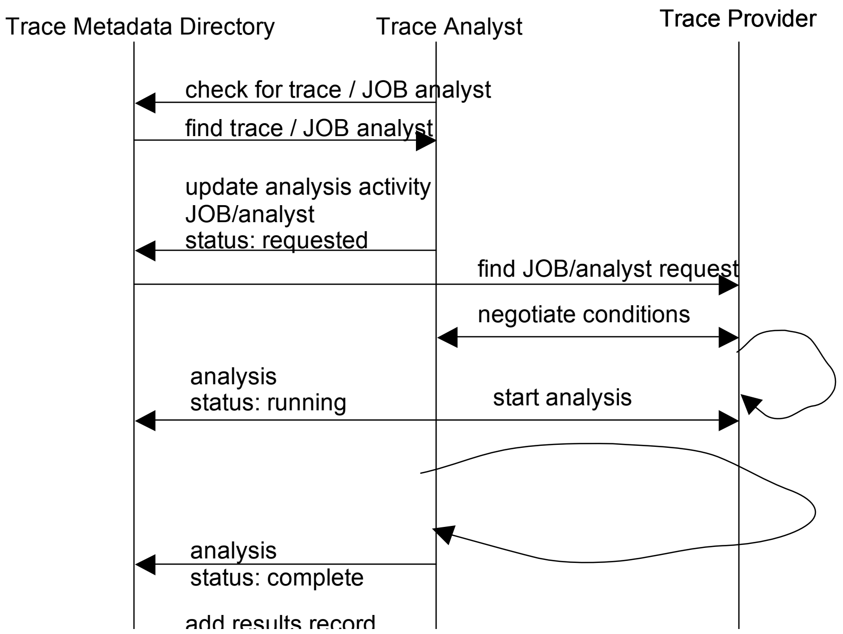


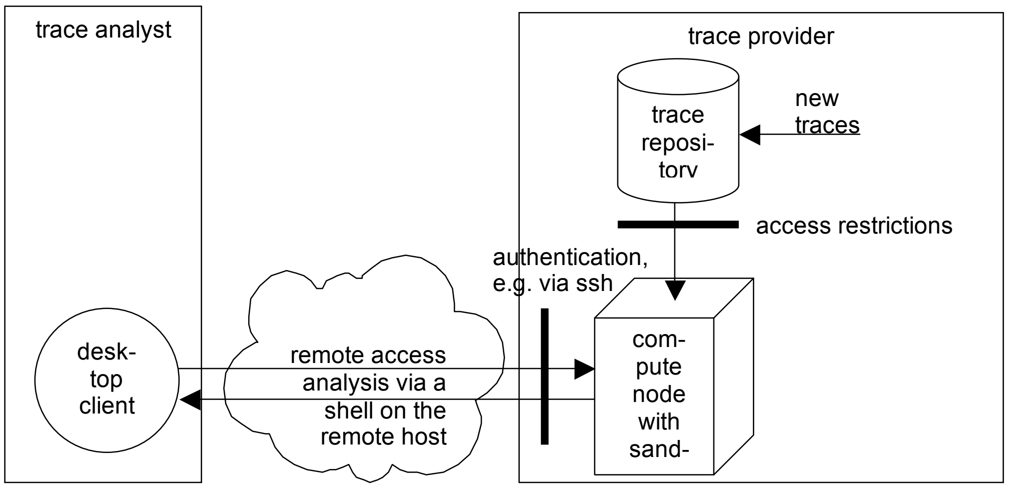
3.4.3.2 *REQUEST_REMOTE_ANALYSIS_JOB_BY_ANALYST*

Informal description:

> "Dear trace provider, we have filed an analysis activity. We would like to run an analysis job on a trace, without the data being copied to our site.
>
> We would like to have remote access to an environment to run jobs on the data."

The analysis job is carried out on the provider side. The code/analysis job is copied to the location where the data is and executed there. This is useful when processing large traces, where copying data is not feasible. The trace provider allows remote access to a sandbox environment for trace analysts to run jobs.




trace analyst
trace provider
trace reposi- torv
new traces
access restrictions
authentication, e.g. via ssh
desk- top client
remote access analysis via a shell on the remote host
com- pute node with sand-

3.4.3.3 *REQUEST_REMOTE_ANALYSIS_JOB_BY_PROVIDER*

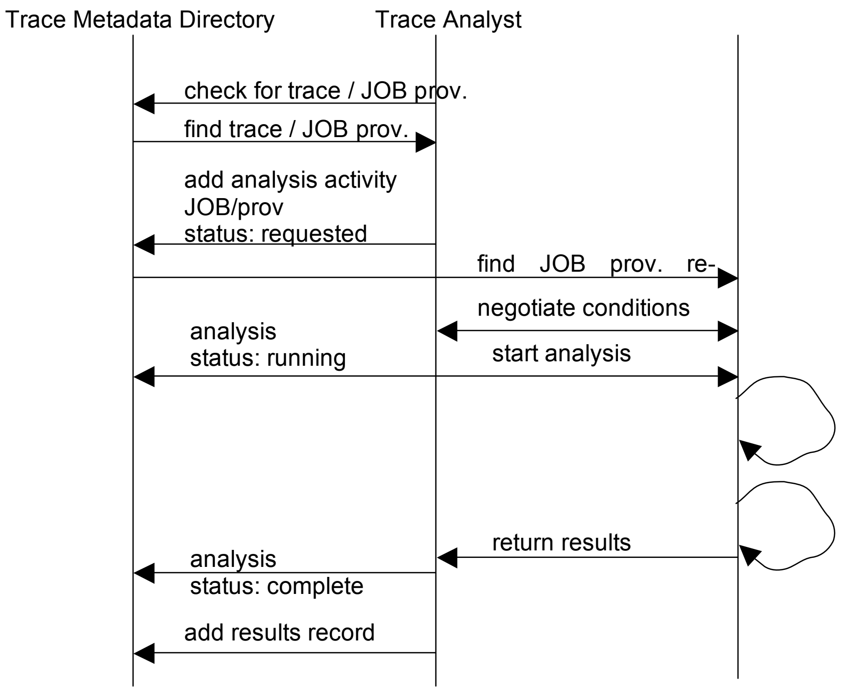
Informal description:

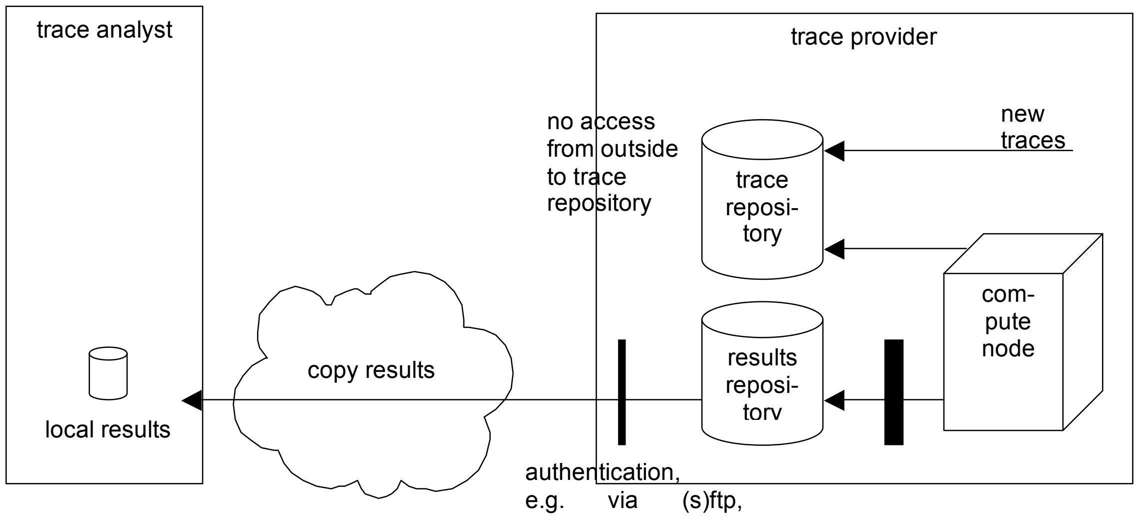
"Dear trace provider, we have filed an analysis activity. We would like have an analysis job run on a particular trace, without the data being copied to our site.

Please carry out the job, that we specified and return/give us access to the results."

The trace provider accepts to receive trace analysts' analysis jobs, that the trace provider will run with own staff without the trace analyst having direct access to trace data. The result is copied to the analyst using a push or pull method.

The trace analyst should specify the job in script form, so that the provider has minimal efforts to actually run the job. A common tool set and runtime environment should be agreed upon.



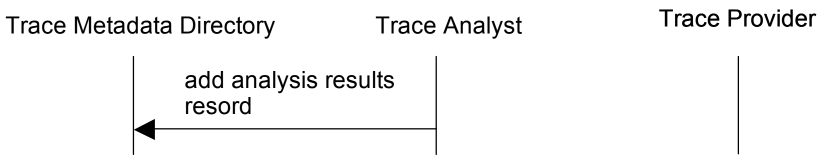


3.4.3.4 *ANNOUNCE_RESULTS*

Informal description:

"We have performed a trace analysis. Here's what we can say about the results."

A result record is added to the metadata directory. The results are thereby announced. We should share us much as possible. Results that required a lot of computation should be shared, instead of being done again and again by each partner. It is necessary to agree on the access control policies that determine which results are accessible to whom.



ANALYSIS RESULTS RECORD

metadata	id	
	reference to ANALYSIS ACTIVITY RECORD	
execution	job processing time / analysis performance	How long did it take on what kind of compute node? -> Help others to get a grip on timing. performance metrics can be GB/time or packets/time
analysis results	textual description	Briefly describe the main findings, so that others can evaluate, whether they could make use of the same results (and therefore ask for access) or not.
	URL	include links to results (optional)
legal	constraints on (re)distribution of results	both, trace provider and trace analyst should specify how others may use the data

Results with non-sensitive data can be collected in a results repository. Nevertheless an analysis results record must be added to the trace metadata directory.

3.4.4 Confidentiality, Trust Models, and Legal Restrictions

Many partners expressed concerns about privacy and distribution of traces provided to EMANICS partners.

3.4.4.1 Anonymization

Anonymization is considered absolutely necessary for legal reasons to enable trace providers to give away or provide access to traces. With **anonymization,** the trace provider obfuscates certain data items, e.g. IP addresses, payload, passwords, or replaces them with blanks or random characters. With **filtering,** the trace provider removes certain data items in a white list (only specific items remain) or black list (specific items are removed) fashion. Certain properties must be preserved however, to not render the trace useless for the trace analyst. Therefore, both sides must agree on anonymization as a result of negotiations.

We differentiate between two forms of anonymization/filtering. *General* anonymization/filtering is applied directly after capture, before providing the trace to any partner – this should include all filtering/anonymization required by law. *Specific* anonymization/filtering resulting from negotiations with a trace analyst – this allows to filter out data irrelevant to the trace analyst.

An effective implementation requires a description of the anonymization acceptable for trace analysts as well as a description of the anonymization required by trace provider. Note that these anonymization profiles must match and ideally there should be an algorithmic way to decide whether the descriptions match. Furthermore, a description of the anonymization performed by the trace provider must be stored as metadata of the trace and tools must exist realize the anonymization based on traces in common formats.

For the descriptions we can start with verbal descriptions. Once we know about suitable anoymization tools, we should refer to more formalized anonymization profiles, e.g., configuration files for the anonymization tools.

3.4.4.2 Trust Models

With all the information in the trace metadata directory, this concept of procedures is based on the visibility of all partners' actions. By this form of transparency, we intend to build up trust among partners.

Good (personal) contacts between providers and analysts are essential, however, especially to get things started. We therefore start with an individual trust model. We assume, that trust is not transitive. We re-evaluate these decisions after partners have gained some experience.

Individual Trust One-on-one agreements between trace analyst and trace provider	**Group Trust** a trace provider allows either noone to run analysis on a trace or allows all TRACE partners to access the trace for analysis
each trace provider has its own repository and cares about authentication and authorization	we could have an NDA, that all TRACE partners sign (the risk of failing to come up

we may want to keep authentication data sync'ed between the different repositories, though	with such is there, but it would hopefully sort out delaing with too many legal requirements) boils down to one general agreement among all trace providers and trace analysts -> would enable a (virtual) EMANICS trace pool/repository

3.4.4.3 Legal Restrictions and Requirements

The current procedures try to put the trace provider in a strong position to encourage trace providers to start sharing/allowing to get access to traces. Control and responsibility over trace data remains with the trace provider. Each trace provider must decide for himself how to use legal agreements to make trace analysts liable for misuse when dealing with the traces.

Some trace providers may be reluctant to provide access to traces due to unknown legal restrictions, which also differ in the various European countries that we live in. It might be useful to prepare short (!) “guideline” style documents for each country where a trace provider resides. (But this may require legal knowledge not readily available to EMANICS partners.) To simplify the construction of legal agreements, it might be useful to collect templates for agreements based on agreements that have already been used by EMANICS partners. Such agreement templates may form suitable building blocks for creating additional legal agreements.

3.4.5 Outlook: Grid Infrastructure for Trace Analysis

Grid computing infrastructures may form a good basis to organize the cooperative analysis of network traces. It is planned to define a mapping of the functional components to a Grid implementation. This requires to wrap selected trace anonymization and analysis tools as grid tools.

As a proof of concept, it is planned to set up a Grid demonstrator (initially two nodes) between LMU and CETIM (LMU and CETIM have agreed to spend one grid compute node each). The demonstrator will be shown to other partners with the idea that other partners develop interest and join in. However, it is anticipated that a more conventional implementation of the trace analysis procedures and the Grid implementation will coesxist for some time in order to gather experience and compare the approaches. If the Grid approach turns out to be successful, it may be possible to migrate all procedures to the Grid implementation.

Below is a short overview of the elements of the Grid trace analysis architecture.

Name	Function	Implementation proposal
Trace metadata directory	information store for all metadata	as database with web front end

Trace Repository	common (virtual) repository for all traces in TRACE	directory in distributed grid file system with grid file permissions
Results Repository	(virtual) repository for analysis results	directory in distributed grid file system with grid file permissions
Job Scheduler	schedule and assign incoming analysis jobs, get rid of explicit data copying	grid job scheduler
Job Progress Monitor		grid job monitoring software and web site
Trace Anonymization Service	service that performs anonymization on traces	wrap existing tool(s) as grid service(s)
Trace Analysis Service	service that runs trace analysis jobs	wrap existing tool(s) as grid service(s)
Compute Node	run analysis on trace data	runs Grid middleware
Authentication and Authorization		create Grid certificates in dedicated CA use Grid methods for authentication and authorization

4 Resource Usage Data Collection (ABLOMERS)

ABLOMERS is a set of open source resource monitoring agents conceived for general purpose network monitoring and in particular for scheduling processes. In fact, our approach improves the scheduling process because in most systems, the computational resource monitoring phase is integrated with the resource selection phase. It implies that for any new scheduling has to be a new monitoring request to all nodes participating into the same Virtual Organization (VO). In our approach, we face the problem by splitting both activities into independent frameworks entrusting the monitoring phase to distributed agents running in an autonomous way. These agents offer computational resources availability information at any time in two standard formats. In addition, they generate statistical resource availability reports. These statistical reports could be used by the resource selection phase to determinate in advance (i.e. by means of a heuristic approach) which resources have higher probabilities to be the optimal selection for any users' request.

4.1 Monitoring Agents Functionality

The monitoring agents have been designed as a set of sub-agents, one per type of resource to monitor. Our design is different to similar approaches in its heterogeneity; it could be deployed in any operative system, provided that SNMP daemons are running. This is because the communication between the agent and the computational platform is via SNMP due to the common availability of such protocol in any network component. Another important difference is their availability to present resources utilization and availability average in a statistical way and at any time. Finally, ABLOMERS has the advantage to be completely flexible to the amount of the resources existing in any node, in the sense that it automatically handles its memory buffer to be as large as the amount of physical devices (hard disks, memory modules, processors, etc.) it has to monitor. Therefore, it could be implemented from simple desktop systems to complex multi-processor servers. At the moment of writing this report, five sub-agents have been developed in five threads running with different interval times. The class diagram of these agents is depicted in Figure 5.1.

The PrincipalAgentDeployer class is in charge of starting single thread per any kind of resources and per any amount of resources. It is very important to be mentioned, due to the fact that many monitoring system fails when they try to handle new "hot-plug resources" that have been added to the system. As we mentioned before, every resource is monitored by independent software threads that at certain lapse of time they start again becoming an infinite cycle. The cycle-timing is defined by local or remote administrators through the parameters of the main class at the beginning of its execution.

4.2 Resources Availability and Status Reports

ABLOMERS presents resource availability and statistical information in two formats. The first one is based on XML documents. It is shown in Figure 5.2, as an example of storage availability information of simple desktop PC with more than one hard disk and more than one partition in at least one of them. In order to generate these reports, the ReportGenerator class is called by the corresponding sub-agent in order to translate the collected data to standard XML formats. This document just gives us resources availability information but ABLOMERS also produce alternative documents, also in XML format, offering statistical information per resource. For instance, we considered important to know in advance the average usability per resource. Therefore, these agents were designed to be able of generating these reports. Both kinds of documents are storage in an internal database that could be accessed by management entities if access rights have been created for them.

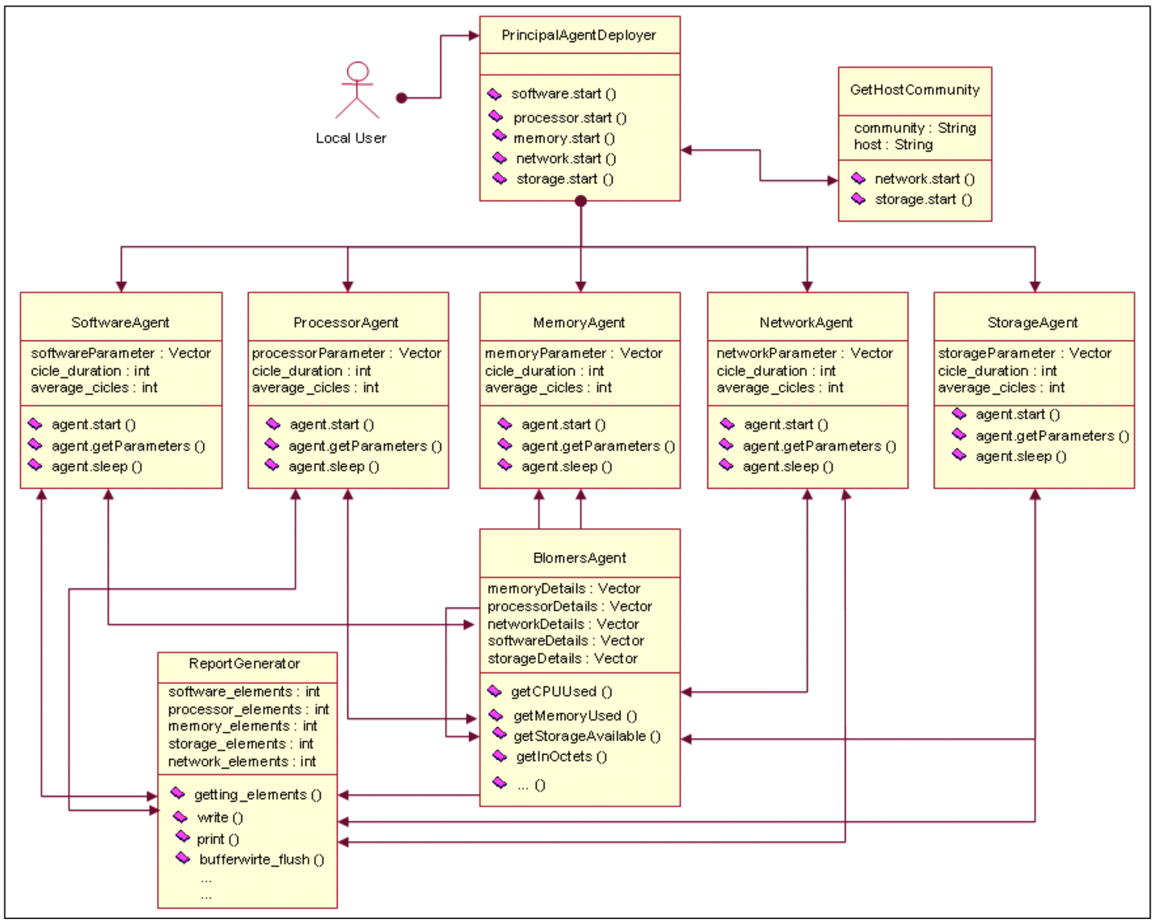
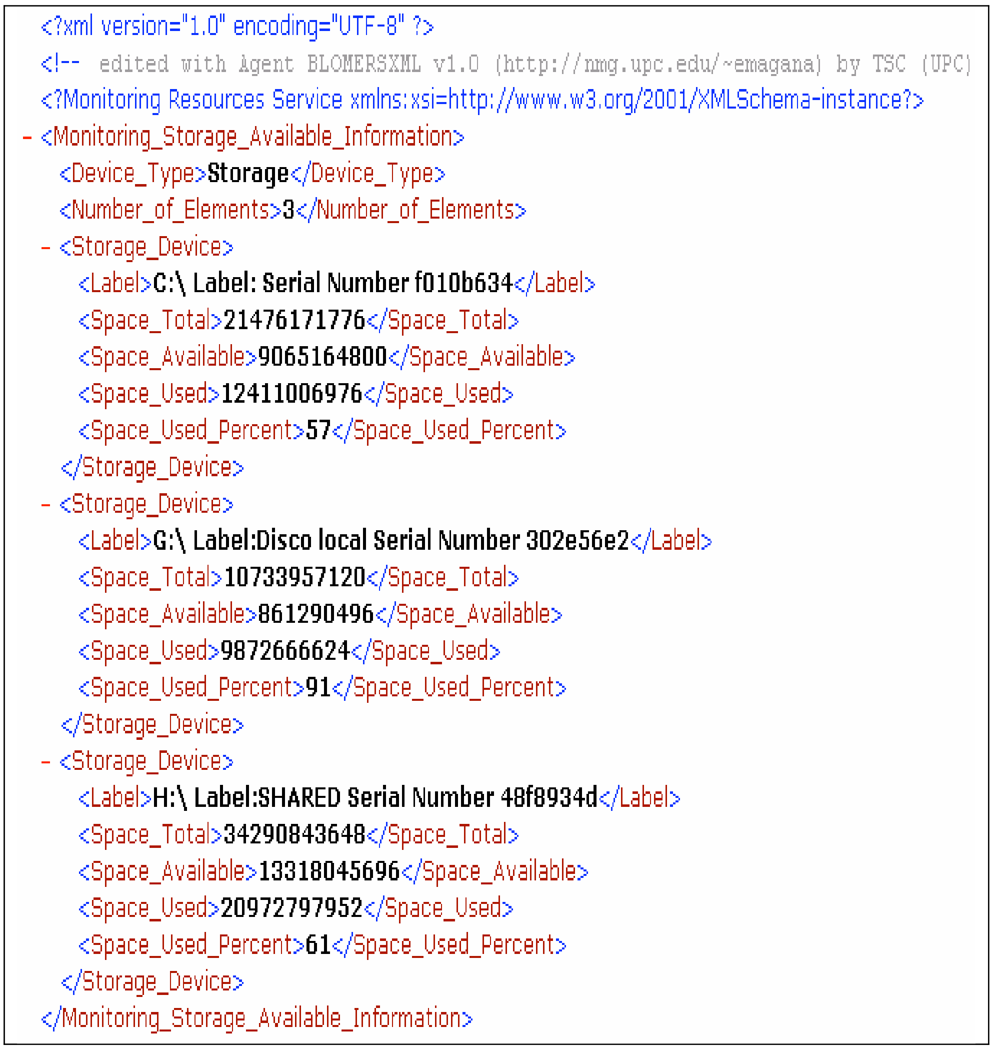


Figure 5.1 Class diagram of the five monitoring sub-agents developed so far

The second format to present monitored information, is through "Vectorial Java Structures", these are structures developed in Java code to keep in memory buffer the amount of resource used and the available ones since the last refresh. Refresh times will be assigned by local or remote node administrator (resources owners) at the moment of starting the monitoring agents. This is another advantage of this approach; a resources manager could get this information from the memory buffer in a faster way than accessing XML files. Moreover, the processor consumption is lesser when parsing activity is not necessary.



```
  <?xml version="1.0" encoding="UTF-8" ?>
  <!-- edited with Agent BLOMERSXML v1.0 (http://nmg.upc.edu/~emagana) by TSC (UPC)
  <?Monitoring Resources Service xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance?>
- <Monitoring_Storage_Available_Information>
    <Device_Type>Storage</Device_Type>
    <Number_of_Elements>3</Number_of_Elements>
  - <Storage_Device>
      <Label>C:\ Label: Serial Number f010b634</Label>
      <Space_Total>21476171776</Space_Total>
      <Space_Available>9065164800</Space_Available>
      <Space_Used>12411006976</Space_Used>
      <Space_Used_Percent>57</Space_Used_Percent>
    </Storage_Device>
  - <Storage_Device>
      <Label>G:\ Label:Disco local Serial Number 302e56e2</Label>
      <Space_Total>10733957120</Space_Total>
      <Space_Available>861290496</Space_Available>
      <Space_Used>9872666624</Space_Used>
      <Space_Used_Percent>91</Space_Used_Percent>
    </Storage_Device>
  - <Storage_Device>
      <Label>H:\ Label:SHARED Serial Number 48f8934d</Label>
      <Space_Total>34290843648</Space_Total>
      <Space_Available>13318045696</Space_Available>
      <Space_Used>20972797952</Space_Used>
      <Space_Used_Percent>61</Space_Used_Percent>
    </Storage_Device>
  </Monitoring_Storage_Available_Information>
```

Figure 5.2 XML Report of Storage Information

4.3 Initial Evaluation Results

We have performed two different evaluation tests. On one hand, we demonstrate that ABLOMERS monitoring agents are not consuming significant computational resources in their hosting node. In order to demonstrate it, we have analyzed the resources' consumption by ABLOMERS agents into a Pentium IV system with 512MB of RAM memory and WindowsXP operating system. The impact on the system performance is almost insignificant. In Figure 5.3, we present the CPU consumption of our agents (dotted line) and the normal performance of the system CPU without any process running in the system (continuous line). It is clear that only in very small intervals of time, system performance could be affected but in general it does not notice any impact. The highest peak during a long performance test is around ten percent (10%) of the total available CPU, this is almost worthless in current systems. In Figure 5.4 we also show memory performance for ABLOMERS. The continuous line shows the memory consumption without agents running in the system and dotted line with these agents. We have found an increase of five percentage (5%) in memory used by the ABLOMERS monitoring agents but most important is that, it remains stable during all performance tests. Therefore, we assure that agents do not affect system performance despite they are running all time (non-stopping).

The second test has been the collection of resource availability information from several personal desktop systems with Linux and Windows platforms for a twenty-four hours period. At the end of the day, ABLOMERS agents keep running and working perfectly with similar performance values such those that have been shown in previous test. We have configured ABLOMERS to generate five different reports, one for each resource to monitor, with different intervals of time between every sensing of the resource availability information. It was decided in this way, mainly because some resources have less rate of variability than others, e.g. the clearest case are software programs, which normally do not change as frequent as memory or processor resources.

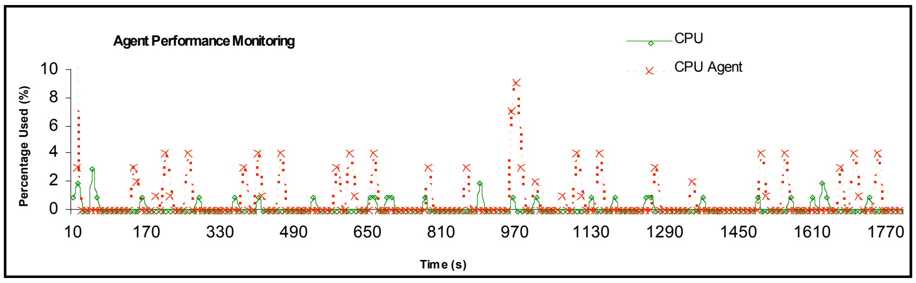


Figure 5.3 ABLOMERS CPU Consumption

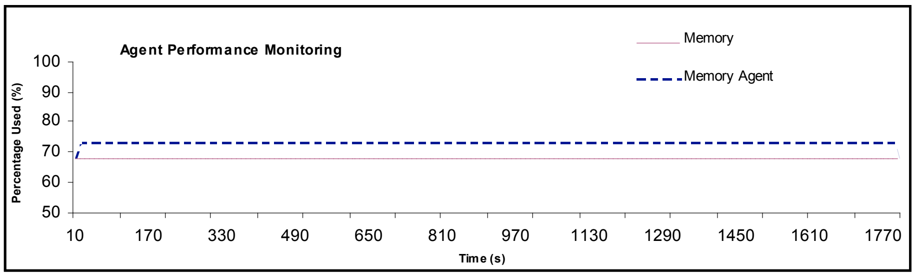


Figure 5.4 ABLOMERS Memory Consumption

In Table 5.1 presents for each type of resource monitored, the sensing interval, the total amount of files generated in the sensing process and the total amount of disk space used. These results are reporting an interval of time for the first 24 hours due to fact that agents automatically clean memory the buffers after this period. Therefore, we avoid the possibility to overflow the system buffer and storage devices with the generated monitoring reports.

Resource	Sensing Interval Time (s)	Total Reports	Space Used (MB)
Processor	10	8640	3,52
Memory	60	1460	0,576
Network	30	2880	1,143
Storage	300	288	0,357
Software	1800	48	0,212

Table 5.1 Storage Space Used for Resource Monitoring Reports.

4.4 Concluding Remarks

Evaluation results are promising in the sense that ABLOMERS is a monitoring tool that is scalable and feasible to monitor resources. Although initially conceived to solve scheduling problems, ABLOMERS can also be used and programmed as a general purpose monitoring tool. Future work is addressed to include not only computational but also networking resources. In addition we plan also to deploy the network in testbeds larger than the dimension provided by the testbeds of currently participating partners (UniS, UT, KTH).

5 Collaboration

The collaboration of partners is an important aspect of this work package. The degree of collaboration can be identified by the number of partners participating in each project as well as the nature of the tasks of the partners in each project.

The VOIP project integrates six partners. The work of partners is to integrate their existing local VoIP laboratory infrastructures into a European testbed. The procedure of integration requires cooperative work among partners. This project has achieved a high level of collaboration during the installation of the testbed. The work package meeting in January will be used to discuss further plans about the evolution and usage of the jointly created testbed and it can be expected that more collaborations on specific VoIP questions can be established.

Similarly, the TRACE project comprises six partners. Sharing trace collection tools and traces repositories demand partners to work together. Moreover, the analysis of traces obtained from different networks requires cooperation between partners and network operators to understand phenomena more precisely. This project also has a high potential to achieve a high level of collaboration. Compared to the VOIP project, however, collaboration is so far happens more on a bilateral basis due to the various roles of the partners and the constraints imposed by confidentiality requirements associate with the traces.

The ABLOMERS project integrates four partners. One partner is taking the leading role as the initiator of the ABLOMERS resource usage data collection project while the other three partners act as a data source. The project may involve some additional partners acting as data sources in the future.

In general, the collaboration of partners achieved in this work package is high and it seems particularly successful to support projects, which require establishing a common shared infrastructure in order to address a given problem space.

6 Summary and Conclusions

The second “Virtual Laboratory Integration Report” presents the implementation of three projects within the work package two.

The VOIP project aims at creating a VoIP communication testbed among EMANICS partners. The basic testbed is up and running, thus opening up opportunities for research activities. The on-going work is to upgrade the testbed security and to expand it to more partners.

The TRACE project is related to trace activities. The preliminary achievements contain the development of new trace collection tools, the collection of traces from different production networks, the establishment of trace repositories built on partner sites, and an initial framework for procedures to support trace data collection and analysis activities. The continuing work concerns collecting more traces, developing more powerful analysis tools, proposing new approaches to analyse extremely large traces, and the sharing trace repositories and trace results. One of the challenges to deal with in the future is to deal effectively with confidentiality requirements of the trace providers.

The ABLOMERS project exploits resource usage data for learning and evaluating load-balancing algorithms on distributed systems. This project is work in progress and could benefit from more partners joining as data providers.

All described projects shows that the collaboration of partners is essential in deciding the success of the projects. With six partners, the first and second projects achieve the high degree of the collaboration among partners through integrating the existing labs into the testbed or collecting traces at different networks. The other two projects with fewer partners also obtain the collaboration of the involved partners through joint work.

This report closes the second phase of this work package with encouraging results. The report also poses few issues: (i) The VoIP testbed should now be utilized thoroughly for research activities; (ii) The demand of good analysis tools is crucial due to the huge amount of traces and their complexity; (iii) Confidentiality issues need further discussion between EMANICS partners in order to come up with effective ways to share data.

During the remaining six months of the first EMANICS phase, this work package will focuses on the continued implementation of the started projects. This requires addressing the issues mentioned above. Furthermore, it is expected that more face-to-face meetings and exchanges will be necessary to effectively support research or teaching activities with the testbed.

7 References

[1] Asterisk, The open source PBX, http://www.asterisk.org/

[2] OpenSER, The open source SIP Server, http://www.openser.org/

[3] J. Rosenberg et al. SIP: Session Initiation Protocol, [RFC 3261], 2002, http://www.ietf.org/rfc/rfc3261.txt

[4] M. Spencer., Distributed Universal Number Discovery (DUNDi), [Internet-Draft], 2004, http://www.dundi.com/dundi.txt

[5] J. Schönwälder. SNMP Traffic Measurements. Internet Draft <draft-irtf-nmrg-snmp-measure-00.txt>, May 2006.

[6] R. Frye, D. Levi, S. Routhier, and B. Wijnen. Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Frameworks. RFC 3584, Aug. 2003.

[7] M. Havan and J. Schönwälder. Prefix- and Lexicographical-order-preserving IP Address Anonymization. In *10th IEEE/IFIP Network Operations and Management Symposium*, Apr. 2006.

[8] K. McCloghrie and F. Kastenholz. The Interfaces Group MIB. RFC 2863, June 2000.

[9] R. Raghunarayan. Management Information Base for the Transmission Control Protocol (TCP). RFC 4022, Mar. 2005.

[10] The EMANICS Web site. http://www.emanics.org

[11] The Simple-Web. http://www.simple-web.org

8 Abbreviation

NoE	Network of Excellence
PBX	Private Branch Exchange
PSTN	Public Switched Telephone Network
RADIUS	Remote Authentication Dial-In User Service
SIP	Session Initiation Protocol
VoIP	Voice over Internet Protocol
SNMP	Simple Network Management Protocol
MIB	Management Information Base
SME	Small and Medium-sized Enterprise
DUNDi	Distributed Universal Number Discovery
PDU	Protocol Data Unit

9 Acknowledgement

This deliverable was made possible due to the large and open help of the WP2 team of the EMANICS NoE. Many thanks to all of them.